Appendix J

SWSC Domain Engineering Lessons-Learned

Contents

J.1 SWSC Domain Engineering Lessons-Learned

The Space and Warning Systems Center (SWSC) domain engineering team presented their lessons-learned from two years experience on the Space Command and Control Architectural Infrastructure (SCAI) re-engineering program in April 1995 at the Software Technology Conference, Salt Lake City, Utah. The SWSC at Cheyenne Mountain, Colorado, maintains and modifies C2 systems for US. Space Command, North American Aerospace Defense Command (NORAD), and Air Force Space Command (AFSPC). These systems are comprised of 26 stovepipe systems, 12 million lines-of-code, 24 different languages, 34 separate operating systems, and numerous proprietary hardware and software components — all having complicated software support environments, as illustrated in Figure J-1. This maintainer's nightmare was fertile ground for architecture-based domain engineering.

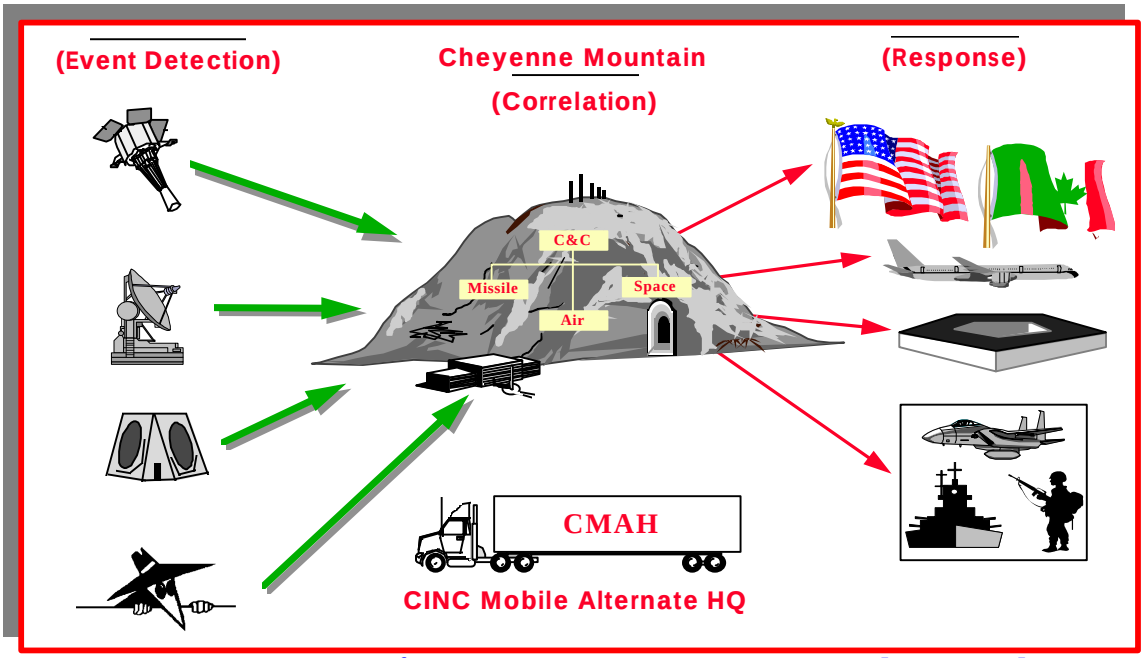

Figure J-1. SWSC Software Re-engineering Environment [BULAT95]

The SCAI Project is using the domain engineering approach developed by the Software Technology for Adaptable and Reliable Systems (STARS) program called *megaprogramming.* They used domain analysis to create a domain-specific architecture to which all individual systems in the domain are mapped to ensure product-line software reuse, as illustrated in Figure J-2.

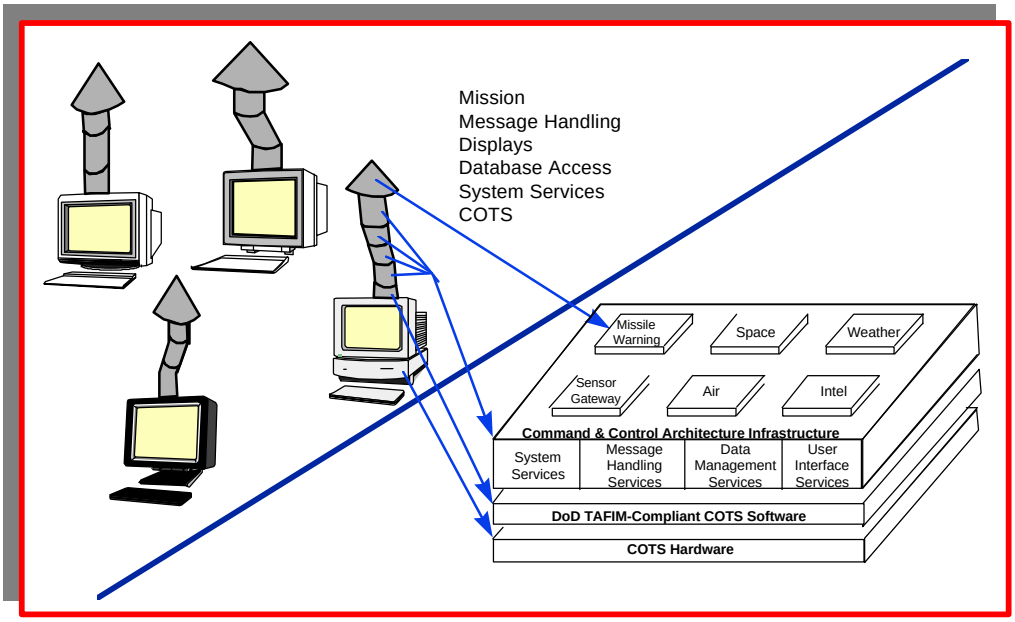

Figure J-2. SWSC Domain Engineering Approach [BULAT95]

The architecture developed for the SCAI program is decomposed into a layered domain requirements model (DRM) and a set of application architectural models (AAM)s. The current scope of the DRM is the SWSC space domain; each AAM is specific to one system in the domain. The layered DRM is a modified Booch-type model, while the AAM is a network topology model and a mapping of application tasks to machines. [BOOCH94] These models comprise the SCAI domain architecture, as illustrated in Figure J-3.

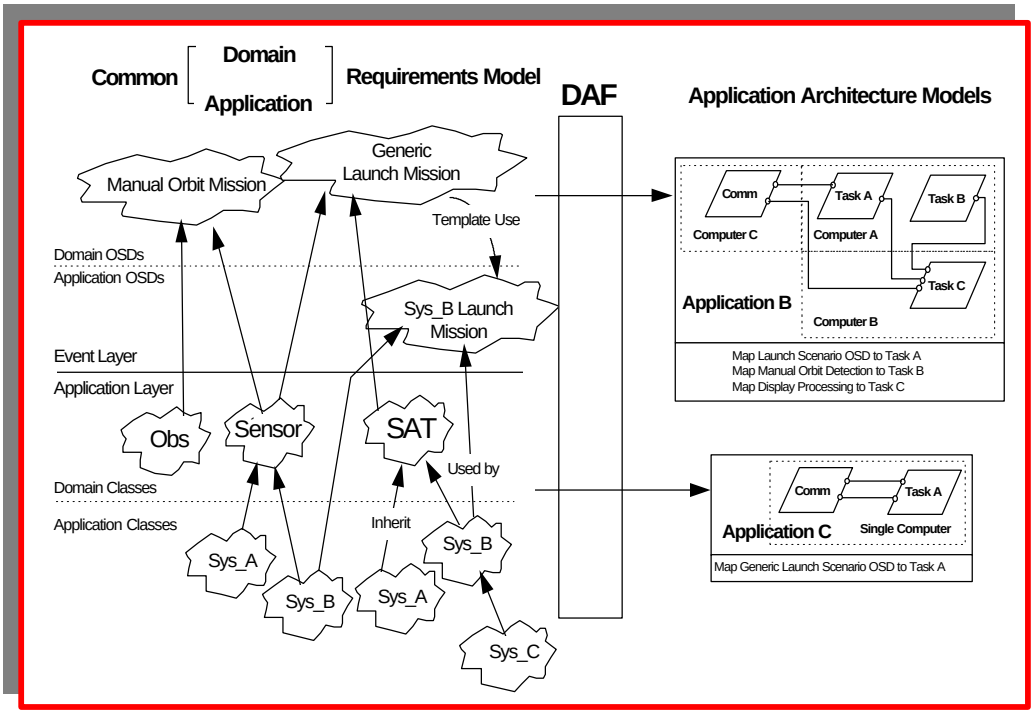

Figure J-3. Domain / Application Model Relationship [BULAT95]

The SCAI is using a product-line software development approach which implements two software life cycles: one life cycle where generalized domain products are developed, and a parallel life cycle where individual applications are constructed from domain life cycle products, as illustrated in Figure J-4. Obviously, domain products must exist prior to their use in the application. The problems the team encountered associated with the requirement for prior construction of all domain components were:

- There were substantial up-front domain engineering costs not associated with developing any product. (As DoD shrinks, these costs are increasingly difficult to justify.)
- Generalized models and generalized components can only be validated through their use on real systems. Even within a single system, a reusable class must be validated in each of the contexts in which it is used. Monolithic *waterfall* systems development has largely been discredited *vis-à-vis* more iterative approaches to modeling and systems development.
- A large domain, such as C2, may contain many complex systems. In spite of the fact that all these systems have much in common, it is unlikely that domain engineering can be initially accomplished within the scope of every one of these systems before the need to deliver the first re-architected system occurs.

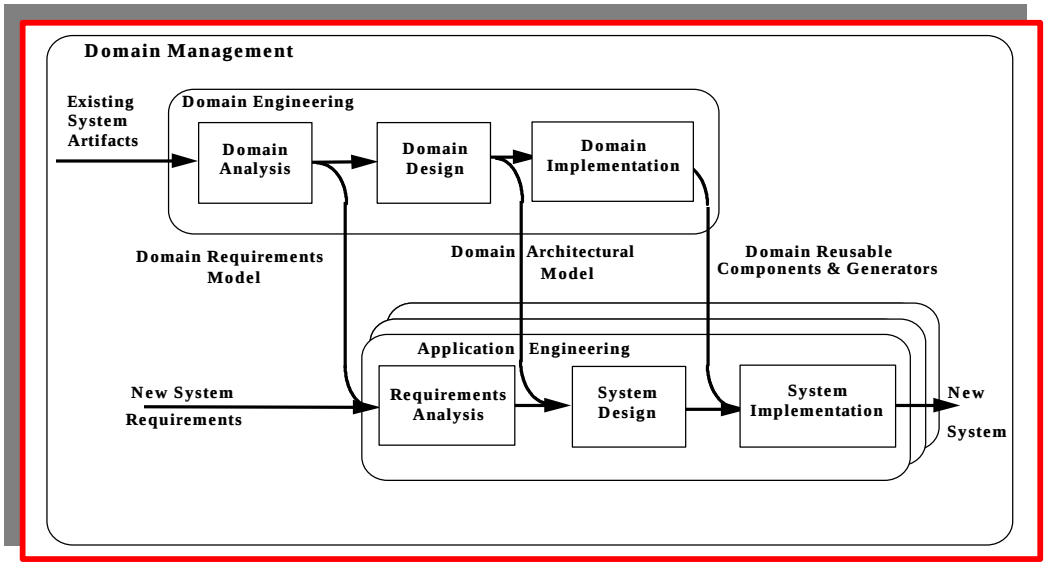

Figure J-4. Iterative Two Life Cycle Domain / Application Engineering Process [BULAT95]

For the above reasons, the domain engineering process must allow for the iterative acquisition of domain knowledge. The SCAI architecture framework was constructed from processes which are inherently iterative; therefore, the overall process is iterative. As new systems are analyzed and the scope of the DRM is extended, more domain missions are identified, new classes are created, and existing classes are generalized. Newly generalized classes are then reinserted into existing missions and retested.

SWSC's goal is to demonstrate that domain engineering will increase software quality, while decreasing the cost of developing and maintaining families of related SWSC C2 systems. As of January 1995, the first intermediate delivery of the SCAI system (the service layer) indicated that over 50% of the code was reused or had been generated. Table J-1 illustrates the program costs for each release (both actual and adjusted) and the amount of program code delivered at the end of each release. It shows the incremental cost per line-of-code, the cumulative program cost, and the cumulative cost per line for each release. The last column shows how initial costs are amortized. As more and more code is developed using the megaprogramming approach, it becomes increasingly cheaper.

SCAI	PROGRAM EXPENDITURES ($Millions)	ADJUSTED COST* ($Millions)	INCREMENTAL LOC	INCREMENTAL COST/LOC	CUMULATIVE COST ($Millions)	CUMULATIVE COST/LOC
Pilot	$2.9	$14.0	125K	$112.0	$14.0	$112.0
Release 1	$3.4	$8.5	267K	$32.0	$22.5	$57.0
Release 2**	$2.6	$6.5	230K	$28.0	$29.0	$47.0
Release 3**	$2.0	$.50	150K	$33.0	$34.0	$44.0
TOTALS	$10.0	$34.0	772K			

* Values adjusted to full life cycle cost plus prior domain work.
** Estimated.

Table J-1. SCAI Cost with Megaprogramming

J.2 References

[BOOCH94] Booch, Grady and Doug Bryan, Software Engineering with Ada, Third Edition, Benjamin/Cummings Publishing Company, Redwood City, California, 1994

[BULAT95] Bulat, Brian G., "Space & Warning Systems Center Domain Engineering Experiences," paper presented to the Seventh Annual Software Technology Conference, Salt Lake City, Utah, 1995